user manual

version 3.2

©2006-2012

CR8 Software Solutions
cr8software.net

Contents

Introduction 3
Technical support and updates

Navigation 4

Getting Started 5
Creating a new font
Common issues
Definitions

View Modes 8
TrueType points view
Nodes view

The Toolbox 9
Select
Pen
Nodes and points
Freehand drawing
Knife
Glue
Ruler
Rotations and other transformations
Lines and shapes
Touch-up tools

The Mapping Window 14
Mapping menu

Glyph Data - .gfs files 16
Copying glyphs between fonts
Saving everything as .gfs

Font Parameters 17
Names
Font metrics
Font description

Composite Glyphs 19
Copy and paste functions
Automatic composite glyph creation

Importing and Tracing Graphics 21
Auto tracing
Manual tracing
Importing vector graphics

Actions 23

Kerning 24

Hinting 25
Alignment zones
Auto-hinting
Gasp (grayscale hinting)

OpenType Features 27
OpenType Layout features
Applying OpenType features
Example

Appendix I. 29
Font Options

Appendix II. 30
Font Basics

Appendix III. 32
PostScript, TrueType and OpenType

Appendix IV. 34
Keyboard Shortcuts

Appendix V. 35
Standard Kerning Pairs

Appendix VI. 36
Action Command Reference

Appendix VII. 38
OpenType feature File Reference

Appendix VIII 41
Composite Wizard File Reference

Copyright 42

Introduction

The art of Typeface design has been around for centuries, and many of the fonts in use today were first designed hundreds of years ago (Garamond typeface, for example was created in the 1500's by a gentleman by the name of Claude Garamond). Back then fonts were cut into steel - today you have the convenience and ease of modern computer software.

Type 3.2 has been designed to make it relatively easy for a beginner to get started making fonts without having to worry about complexities. This version, for example, integrates an autotracing feature (see page 21), making it easy to create fonts from scanned images and graphics. Advanced features are available if required: kerning, automated action scripts, hinting and OpenType features. If you are a graphic artist then you should also find that the drawing tools are not too dissimilar from those found in the likes of Adobe Illustrator and CorelDRAW.

This manual is intended to be an instruction manual for Type 3.2 rather than a tutorial or introduction to font design. If you are new to making fonts then I recommend that you read Appendix II & III of this manual first, and also check out the links below.

Allan Murray
Senior Developer
CR8 Software Solutions

For more information:

Visit **http://cr8software.net/truetype.html** for tutorials, articles and links about making fonts (general as well as specialised Type 3.2 articles).

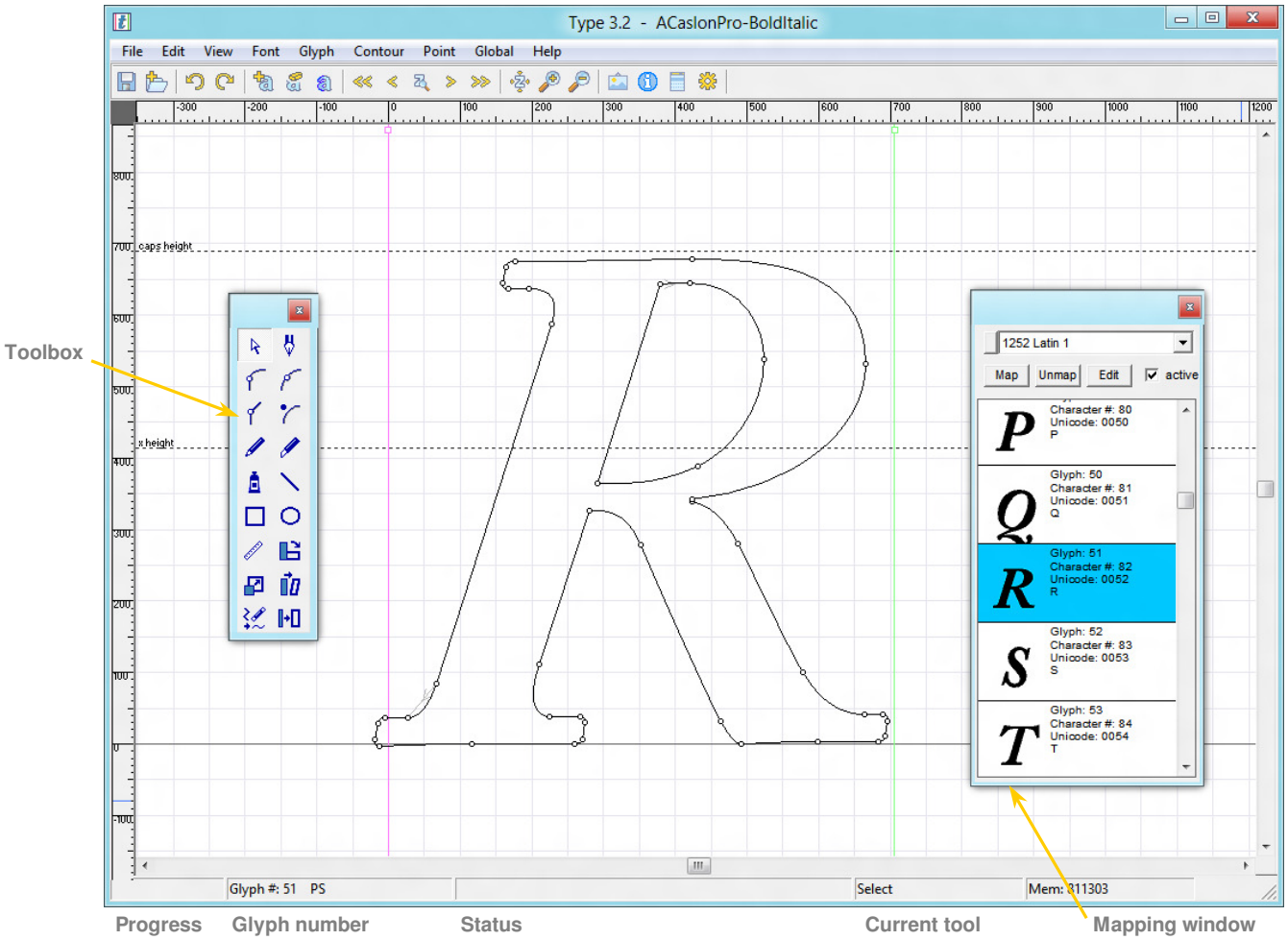
There is a public discussion forum here: **http://cr8.proboards.com** where you can post questions or suggestions, or just to see if someone has already asked your question.

For free support post your private query here: **http://cr8software.net/support.html** or send an email to: **support@cr8software.net**.

Updates:

Free updates are issued from time to time with minor feature additions, improvements and bug fixes. The most current version number is always displayed on the Type 3.2 web page: **http://cr8software.net/type.html**.

To obtain an update, reinstall the software using the download link that you received with your original purchase. You will not normally have to re-enter your registration details.



Navigate around the edit window (main display area), by using the **scroll bars** or by **right clicking** on a clear area to grab and drag the window.

Zoom in and out, using the **magnify** and **reduce buttons**, the **+ key** and **- key**, or the mouse scroll wheel.

The magenta (p1) and green (p2) vertical lines mark the width of the glyph (see Appendix II). Move these by dragging at the top of each line, or select **glyph metrics** from the **glyph menu**.

The dashed horizontal lines are: **ascent**, **descent**, **caps-height** and **x-height**. Anything that is not between the ascent and descent line will be 'clipped'. The values of these parameters can be set using **metrics** from the **font menu** (WinAscent, WinDescent, Caps height and x-height). Do not confuse ascent and descent with typographic ascender and descender mentioned in Appendix II).

Creating a New Font

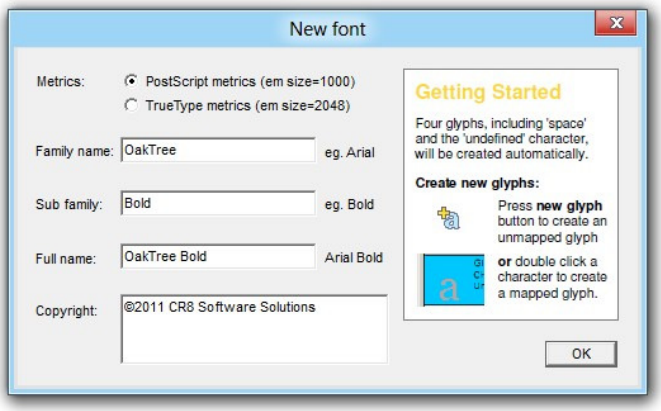
Select **new** from the **file menu**. You will be prompted for the names of your new font. All fields are required. These are:

Family name: Arial for example

Sub family: Regular, Bold, Italic etc.

Full name: Family name + Sub family (eg Arial Bold). For a Regular font, just put the Family name (Arial)

Copyright: Your copyright details.



The first four glyphs will be automatically created and mapped (see Appendix II for more details). The first glyph always represents the **undefined** character – normally an empty box-like symbol. You can edit the symbol, but you cannot map this glyph to anything (mapping to the undefined character is equivalent to unmapping).

The next two glyphs are mapped to certain **control characters,** and the last is mapped to the **space** character. This glyph must be left blank, but you can change its width to match the spacing of your font.

Press the **new glyph button** to start creating a new font.

or

Double click on one of the characters in the **mapping window** and say **yes** to the prompt – a new glyph mapped to that character will be created.

You may now proceed to create glyphs using the range of tools available in the **toolbox.**

Common Issues

The following are a list of tips to help you avoid some common issues that can occur when creating or editing fonts.

- Do not open font files for editing from the Windows font folder. Copy them to another folder before opening them.
- Don't save font files directly into the Windows fonts folder (they will not be installed correctly).
- The correct way to install fonts is to drag the font file (save it somewhere else first) into the Windows font folder.
- If you are editing an installed font, then you will need to uninstall the original or give the new font a different font **family name** before you can install the new font.
- Make frequent backups (always good practice) of your font files, preferably as .gfs files, during the font development process.
- When a font does not work as expected, it is often a **naming** (conflicting font names) or **encoding** issue: make sure **codepage Latin 1252** is marked **active**, and mark any unicode scripts that contain glyphs as active.
- You should only have up to four different fonts that have the same font **family name** –regular, bold, italic and bold italic.
- If you change the name of a font, make sure that you change the advanced names also. Some programs will use advanced names (eg unique font name) to distinguish fonts, so they need to be unique.
- If you change the name of a font, make sure that you change the names for Macintosh (Roman) (otherwise the font may not work if installed on an Apple Mac) and for other active languages.
- Avoid creating glyphs with overlapping contours. The glyph will display correctly on a TrueType (.ttf) font (but is not recommended) but will have a white space in the overlapping region on a PostScript (.otf) font.
- When creating glyphs where one contour is enclosed by another (eg. the inner and outer contours of an 'o'), the inner contour must be in the opposite direction (clockwise or anticlockwise) to the outer contour (see appendix II).

Definitions

Glyph – Glyphs are the shapes and symbols that you design. They normally represent characters or components of characters. A font contains a list of glyphs indexed by a glyph ID. They can be in an arbitrary order, but creating them in character set order is normal practice, and has some advantages.

Character – Characters are the basic symbols that are used to represent a language. The letter A is a character in the Latin Alphabet for example.

Mapping – Mapping is the process of connecting glyphs to characters. Glyph ID #36 may be assigned to represent the letter A character in a certain font for example. A single glyph can be mapped to more than one character, and (using OpenType features) several glyphs can also be used to represent the same character.

Unicode characters – Unicode is an international character encoding system that assigns a code to every character for most of the worlds language systems. A **Unicode script** is a range containing related characters. For example the Basic Latin script is the unicode range 0000 to 007F.

Character set – A Character set is the group of characters used to represent a particular language. Single byte character sets (**Code pages**) can contain up to 256 characters. The characters can represent different languages depending on the encoding used. Fonts can contain more than one Code page, which can be selected from the font selection menu on most programs. (Western = Latin 1252) .

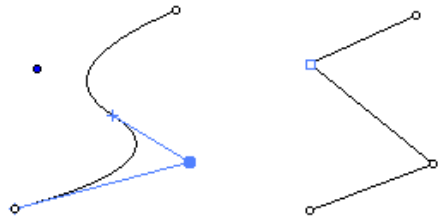
TrueType Points View

Select **TT points** from the **view menu**. This is the native TrueType format and can only be used with TrueType curves (see Appendix III). A TrueType font stores glyphs as a series of contours made of points – either off-curve (dark blue dot/light blue dot when selected) or on-curve (white circle/light blue square when selected).

The gray arrow between point one and point two on each contour indicates the contour direction (see Appendix II regarding correct contour direction).

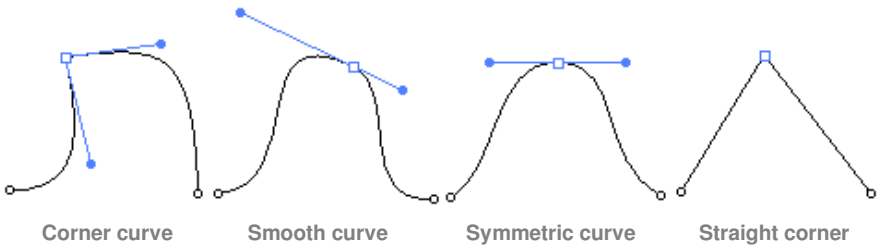
Point one must always be an on-curve point.

Using on-curve and off-curve points is the correct representation of a TrueType glyph, but it is not necessarily the easiest way to design glyphs.



Nodes View

Select **nodes** from the **view menu**. Another way to represent a glyph is by using nodes (white circle/light blue square when selected) which are always on-curve, and control points (light blue dot) which are always off-curve. The light blue line between the control point and the node is at tangent to the curve. Move the position of the control points to change the shape of the curve.*



Corner curves have control points that can be moved independantly (unlinked). Smooth and symmetric curves have linked contol points that are always in line with the node. Straight lines do not have any control points.

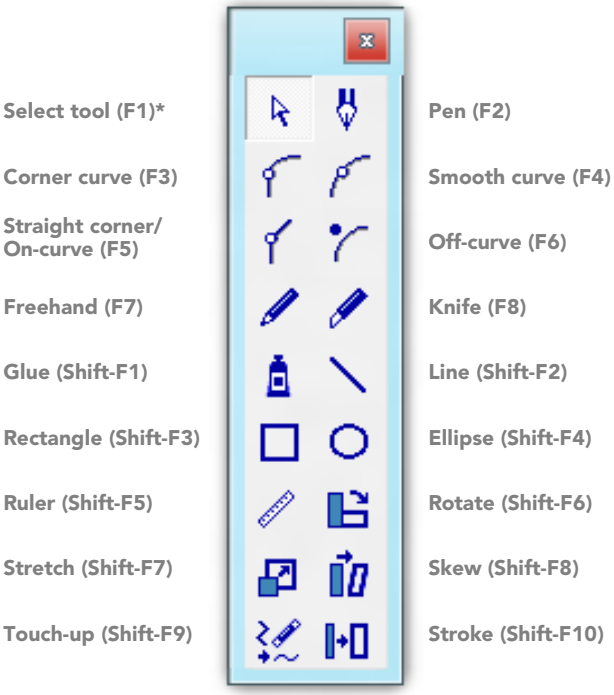
Use the point menu (or right click) to change the type of curve.

* Other font editors only allow this type of editing with PostScript curves – Type 3.2 also allows you to edit TrueType curves in this manner, but limits are imposed to restrict the outline to a TrueType curve. ***Because of these limits, the controls may not behave as you may expect when editing TrueType curves in this mode - nodes other than the one you are moving, may also move, for example.***

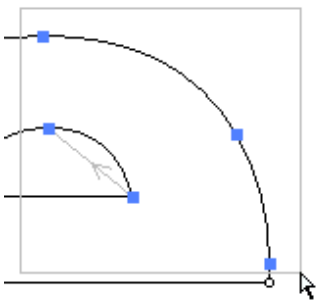
The **toolbox** contains the tools that you will need for creating and drawing glyphs.

Using different tools, you can move points and contours, draw and manipulate lines and curves, create shapes, and measure distances.

*The keyboard shortcut keys for the Toolbox are the function keys F1-F10.



The Select Tool



You can use the **select tool** to highlight and move points around.

Drag a rectangle over points or nodes to select multiple points. Pressing **shift** at the same time allows you to select more points, either by drawing another rectangle, or by clicking on individual points. Pressing **ctrl** at the same time as selecting points will select the entire contour.

Multiple point selection allows you to move many points (move one of the selected squares to move all of them, or use the cursor keys), perform transformations (see below) and allows other contour operations using the **contour menu**.

Selecting a point by clicking on it allows you to move a single point, and gives you access to the options on the **points menu** – either from the main menu or by clicking the right mouse button.

If you are in **nodes view mode** then when you select a node, the control points belonging to that node will be displayed. The control points (light blue dots) control the curve - click and drag them to alter the shape of the curve. Control points can be removed by dragging them into the node, and then releasing the mouse button (PostScript curves only). Clicking on a node while pressing **shift** allows a new control point to be dragged out of the node (PostScript curves only).

When editing a glyph with TrueType curves, moving a conrol point may also affect neighboring nodes. PostScript curves do not have this limitation.

The Pen Tool

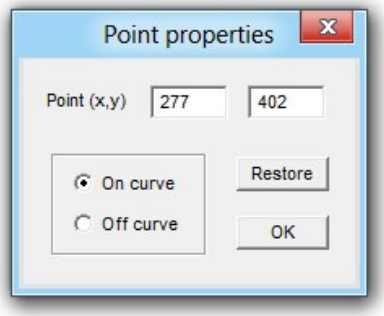
The **pen tool** will draw smooth curves. Left mouse click to add a node, and drag the mouse before you release the button to extend the control points. The curves will be symmetric about the node. If the **shift** key is pressed then the control points will no longer be linked, and you will be able to form smooth or corner curves.

If you are editing a glyph with TrueType curves, then when you release the button the curve will convert to a TrueType curve (with additional nodes).

The Node and Points Tools

These four tools work as follows:

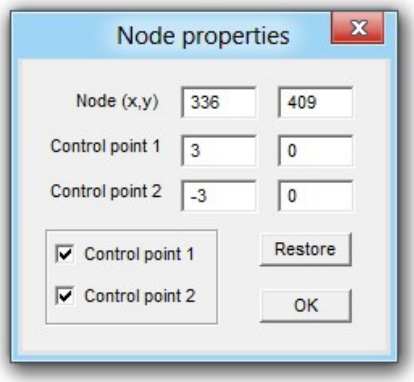
- If no point or node is currently selected, clicking will start a new contour.
- If the last point or node on a contour is selected, then clicking will add a node or point to the current contour.
- Clicking on the first point or node of a contour will close the contour.
- If the mouse pointer is over a curve or line, then a small box will appear next to the mouse pointer indicating that clicking will insert a new point or node in the middle of the contour.



The **corner tool** and **curve tool** allow you to adjust the degree of curve. If you move the mouse before you release the button, you can change the position of the control points.

The **off-curve tool** functions only when editing a glyph with TrueType curves.

Points and nodes can be precisely positioned by using **properties** from the **points menu** (or right click, **proporties**).

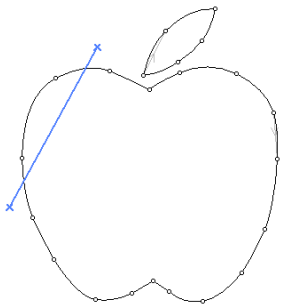


Freehand Drawing Tool

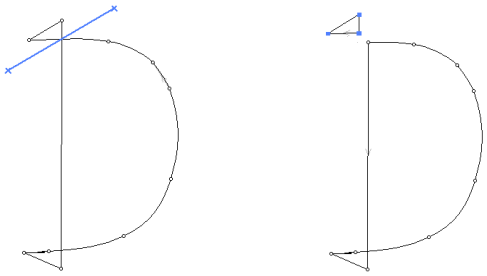
You can draw free hand using the **freehand tool**. When the mouse button is released, the shape drawn will be converted to nodes and contours. Draw slowly for best results. You can also pair this with the **stroke tool** to add 'thickness' to unclosed contours that you draw.

The Knife Tool

The **knife tool** can be used to 'chop' contours in half – draw a line through the contours where you want to make the cut. Pressing the **ctrl** button at the same time will insert nodes, but will not cut the contour.

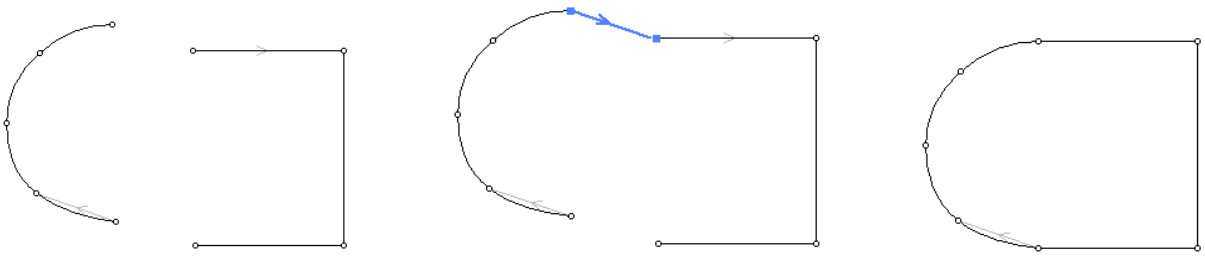


This tool is useful for removing overlapping contours that may occur when a glyph has been over-thickened with the stroking tool. Cut the overlap with the **knife** tool, then join the remaining ends with the **touch-up: connect** tool:



The Glue Tool

Use the **glue tool** to join contours together. Click on a node, then drag to a node on another contour. Release the mouse button, and the two contours will be joined. Beware that contour direction is important here, and that the direction of the contours remains unaltered.
(To change the direction of a contour, use **reverse** from the **contour menu**).

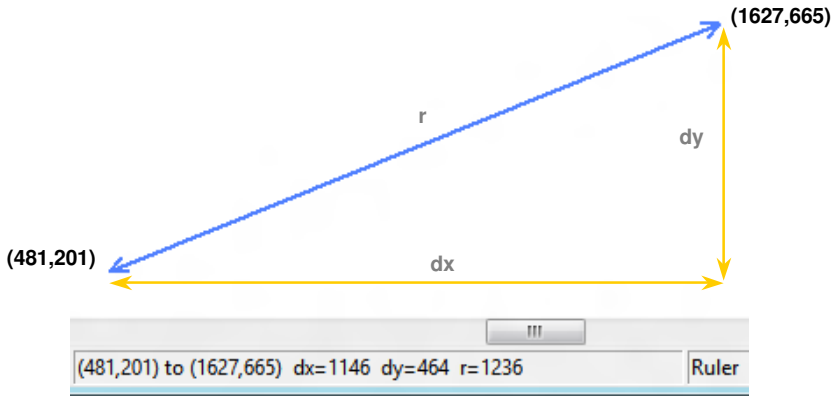


1 2 3

Pressing the **ctrl** button at the same time will allow you to glue the last node of an open contour to the first node of another open countour using a connecting line, without moving either contour.

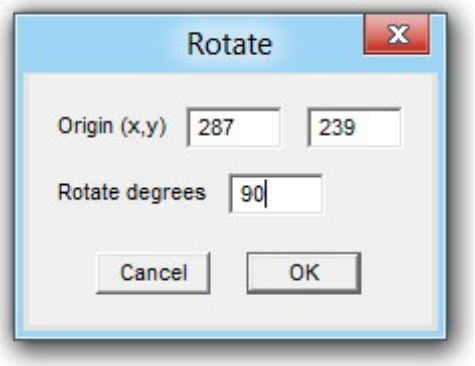
The Ruler

The **ruler tool** can be used to measure distances. Click at the start position (or node) and drag to the end position (or node). Various measurements will appear in the status bar:



Rotation, Transformation and Stroke Tools

The three of transformation tools and the stroke tool all work in a similar way.
There are two options:



1. Visually transform using mouse control

Left click at the origin* for the transformation, move the mouse until the selected points are in the correct position, then release the mouse button.

2. Precise keyboard input

Right click at the origin, then enter the exact parameters.

In both cases, the origin will be marked by a yellow cross.

- Only selected points (or selected contours for the stroke tool) will be altered.
- If no points are selected, then the whole glyph will be transformed.
- There is a limit to how much a composite glyphs can be transformed and the stroke tool cannot be used on composite glyphs (use the decompose button first).
- After using the stroke tool the glyph will be converted to PostScript curves.

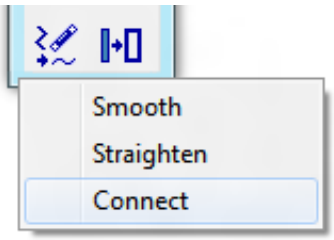
*No origin is required for the stroke tool, so it does not matter where you click.

Drawing Lines and Shapes

Lines, rectangles and ellipses (or circles) are easy to draw by using the **line tool**, the **rectangle tool** and the **ellipse tool**. Click on the start position and release the mouse button at the end position.

- If a line starts at the end of a contour, it will be joined to that contour, and if it ends at the beginning of a contour, the contour will close.

The Touch-up Tools



The **touch-up tools** are one-off operations performed on a group, or groups of selected points or nodes. The current tool can be changed by right clicking on the **touch-up tool** button and selecting a new tool.

Smooth tool

This tool can be used to create smooth curves between points. It is useful for smoothing curves created using the **freehand tool** or graphics that have been autotraced.

- After using the smooth tool the glyph will be converted to PostScript curves.

Straighten tool

This tool will remove curves from the selected points:

- If in nodes view mode, then the control points will be removed from selected nodes.
- If in TT points mode, then this operation will convert any selected off-curve points to on-curve points.

Connect tool

This tool will connect selected points if they are in close proximity to each other. It is used to close or connect contours – similar to the glue tool – but allows joining without moving contours or inserting extra paths.

- One of the points must be the first node of a contour and the other point must be the last node of the same or a different contour.

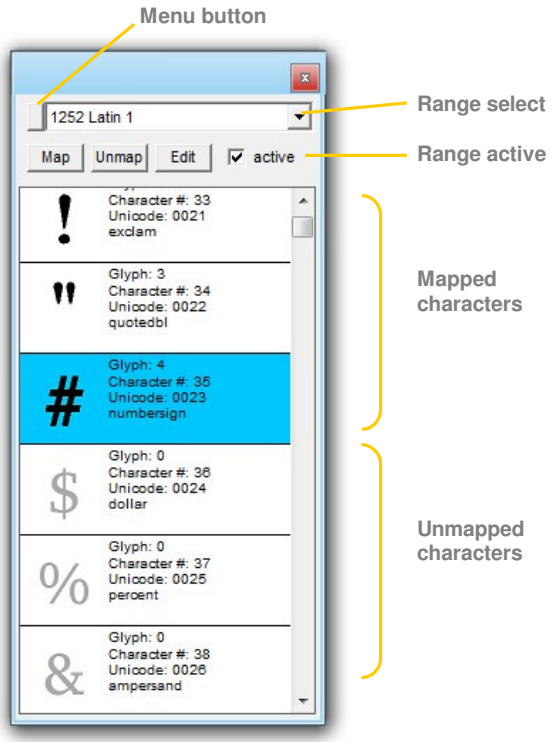
The points to be joined must be in close proximity (several pixels).

Mapping Window Modes

The **mapping window mode button** will sequence through the different modes available. You can also change the mapping mode using **preferences** from the **view menu**. The modes available are:

1 - Code Pages (character sets)
2 - Unicode Scripts (a range of unicode values)
3 - Glyph List (all mapped and unmapped glyphs)
or the glyphs contained in an external font

Use **range select** to select which characters are displayed, and the **range active box** to indicate which ranges are used in your font. To make a Western character set font, for example, select the **1252 Latin 1 page range**, and map glyphs to these characters, then check the **range active box**.



Mapping Glyphs

Modes 1 and 2 allow glyphs to be mapped to unicode characters (uni0000 to uniFFFF). If the **new glyph button** was used to create the glyph, then it will need to be mapped:

To map the current glyph (the glyph in the edit window) to a character, select the character in the mapping window (it will become highlighted blue), then press the **map button**. A single glyph is normally mapped to a single character, but may be mapped to multiple characters.

Multiple glyphs can be mapped to multiple characters in one operation. Select the first character to be mapped, then, while pressing **shift** or **ctrl**, select the last character. The whole range of characters will be highlighted blue. If, for example, glyph #10 is the current glyph, then it will be mapped to the first character, glyph #11 will be mapped to the second character, etc, when the map button is pressed.

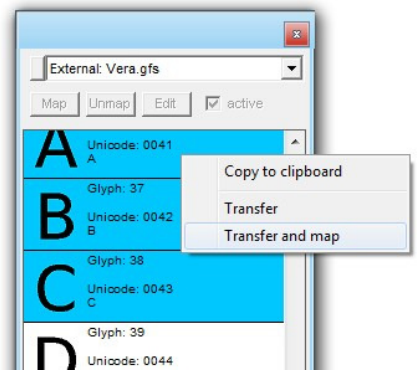
Modes 3 (glyph list) allow glyphs to be mapped to all unicode characters, including characters in the supplementary planes (uni10000 to uni10FFFF). Select a glyph or range of glyphs in the mapping window, then press the **map button**. A prompt will appear. Enter the unicode code point (in hexadecimal format) that you want to map the glyph (or first glyph in the range) to.

Mapping Menu - Bookmarks

Press the **menu button** will access to the **mapping menu**. Here you can add or remove **bookmark buttons**. A bookmark will allow you to easily return to a specific position in a script, codepage or glyph list. Bookmark buttons will be displayed in the main toolbar. To remove a bookmark button, press the bookmark button, then select **remove bookmark** from the **mapping menu**.

Mapping Menu - External Fonts

Use **add external font** from the **mapping menu** to list an **external .gfs font file** (partial or complete - see next section) in the mapping window. Select a glyph or range of glyphs, then right click on the selection to transfer them to the working font. The next section (Glyph Data - .gfs files) explains the options available for transfer.



Resizing

Some glyphs (composite glyphs or glyphs of some non latin alphabets) may not fit into the default row size of the mapping window. To resize the rows:

1 - Make sure **enable row resizing** is checked in **preferences**.

2 - Select a row in the mapping window (highlighted blue).

3 - Widen the character space by clicking and dragging where the text margin is located.

4 - Increase the height of the character spacing by dragging the base of the highlighted cell.

Symbol Fonts

A **symbol font** is a special type of font normally containing pictorial type glyphs. Symbol font glyphs are mapped to unicode F020 to F0FF (corresponding to ASCII characters 20 to FF).
To create a symbol font, map glyphs to the characters in the **Symbol Character Set** Code Page and set to **active**. NOTE: Setting the Symbol Character Set to active will overide any other active Code Pages, and only characters F020 - F0FF will be mapped in the output font.

Type 3.2 supports it's own font data file (.gfs). A .gfs file can contain a range of glyphs (a partial font file) or the entire font including all supporting tables and information (complete font file).

Saving Everything as a .gfs Font

You can use **save .gfs data** from the **file menu** to **save everything as .gfs font.** The entire font will be saved, including mapping, metrics, composites, kerning and other tables. This file can be re-opened at a later time using **open .gfs data** from the **file menu.** This is the best way to save a font during development and as a backup - especially with very large fonts.

- Keep a working copy of your entire font in **.gfs** format, then export your final font to **.ttf** or **.otf** when required.

Copying Glyphs Between Fonts

You can use **save .gfs data** from the **file menu** to save a range of glyphs to a **.gfs file**. You can then add these glyphs to another font using the **add .gfs data** option. The glyphs will be added at the end of the working font.

- When adding glyphs from a partial font file, composite glyphs and mapping information will not be transferred.

The **add .gfs data/ open .gfs data** option works as follows:

- If a font is open, then Type 3.2 will append a **partial font file** to the end of your font.
- If there is no font currently open, then Type 3.2 will allow you to open a **complete font file**.

Transferring Glyphs from an External Font

External fonts that have been loaded into the **mapping window** for viewing (see previous section) can be transferred to the working font. Mapping can also be transferred. The options are:

copy to clipboard: The first selected glyph will be copied to the clipboard.

transfer: The range of glyphs selected will be added to the end of the working font.

transfer and map*: Glyphs will be transferred and mapped. If the working font has a glyph already mapped to the same character, then that glyph will be replaced with a glyph from the external font, otherwise the transferred glyphs will be added to the end of the working font.

transfer and merge*: Same as transfer and map - but existing glyphs are merged, not replaced.

An action script may be required to scale transferred glyphs to a size to match the working font.

* These options are only available when using .gfs files that have been created using Type 3.1 or later.

An OpenType font contains a whole lot of information – names, font descriptions, parameters and metrics specific to your font. Type 3.2 allows you to set and change most of these.

If you are new to font design, and are creating a basic font, then you can keep the default values and use the automatic settings. See Appendix II for a better understanding of various parameters.

Names

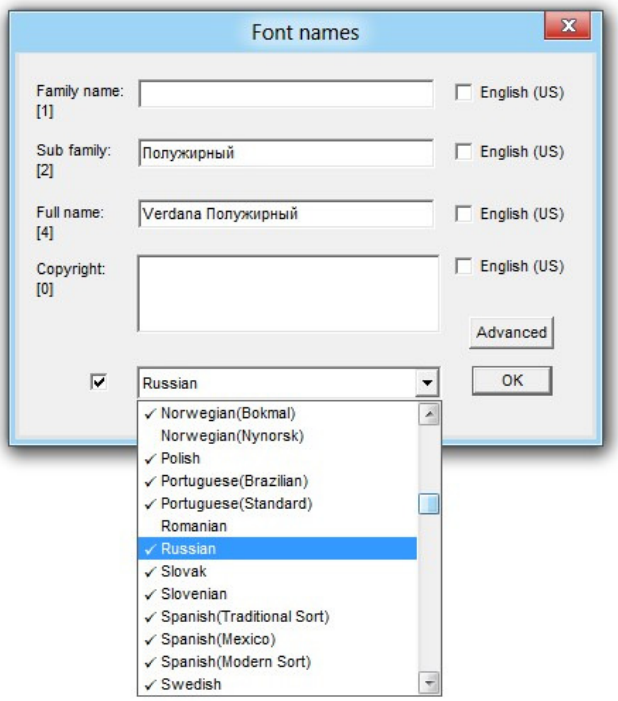
When you create a new font, you are prompted for certain name information. This information is for the default language – English (US). Your font can contain name information for other languages too. Use the **names** from the **font menu** to add more entries to the names table.

Select the language at the bottom of the window and check the box to the left to make the language active. Blank fields will not be listed in the names table . If a English (US) box is checked, then the English (US) string will be duplicated in that names table listing.

The **advanced button** allows you to enter various other names. The most important ones are: **Unique ID [3]** and **PostScript name [6]**. If you change the name of your font then you must change these also – or blank them out. If these fields are invalid or blank, then Type 3.2 will generate valid names when you save the font.

For an ordinary Latin font, name information would normally only entered for **English (US)** and **Macintosh (Roman)** languages (these two must always be present).*

Name strings support unicode, so you can paste unicode characters into the naming window input boxes.



* Most non Latin fonts actually only have enteries for English (US) and Macintosh (Roman).

Font Metrics

Use **metrics** from the **font menu** to set your font's metrics. These parameters are certain measurements specific to your font. Some of these are self explanatory like **underline thickness** and **italic angle**. The most important ones to understand are:

EM Unit size:	Usually set to 2048 (TrueType or OpenType TT), or 1000 (OpenType PS). This is used to calculate the point size when the font is displayed.
WinAscent:	top (anything above this may be clipped)
WinDescent:	bottom (anything below this may be clipped)
Caps height:	uppercase character height
'x' height:	lowercase character height (height of a lowercase 'x')

If you have checked the option **show horizontal markers** (using **parameters** from the **view menu**) then the last four parameters above will be visible in the main edit window as horizontal lines.

Font Description

Use **description** from the **font menu** to set the parameters that describe the font: weight, width, version number, italic, bold and mono-spaced. If you set the font to **Mono-spaced** then the advance width of all glyphs (except glyph number 1 - which must have an advanced width of zero), will be constant. Advanced description parameters can only be altered using Type 3.2 full version.

A composite glyph has no contours or points of its own, but is made up of other glyphs. Creating a composite glyph is as simple as copying the component glyphs into a blank glyph. The composite is displayed as a blue outline, and although you can move and transform the components (as if you were manipulating contours) you cannot move individual points unless you first **decompose** the glyph. The **decompose button** is sixth from the left on the toolbar.

Glyphs can be stored as composites in a **TrueType font (.ttf)**, but for an **OpenType PostScript font (.otf)** the glyphs will be not stored as composites – they will be decomposed. (Save a copy as a **.gfs file** if you want to alter the glyphs as composites at a later time).

Copy and Paste Functions

- Use **copy glyph** from the **edit menu** to copy the entire current glyph to the clipboard.
- Use right click **copy glyph** from the **mapping window** (glyph list mode only) to copy the selected glyph to the clipboard.
- Use **copy** from the **edit menu** to copy only the selected points to the clipboard.
- When the destination glyph is in the Glyph Window, use **paste** from the **edit menu**. When copying glyphs into a blank glyph, a composite glyph will be created. Press the decompose button if you don't want this.
- The **paste metrics** option from the **edit menu** pastes only the Left Side Bearing and Advance Width from the clipboard.

Automatic Composite Glyph Creation

The **composite glyph wizard** can be used to generate characters with diacritics (eg accented Latin characters). There are three in-built ranges of glyphs that can be generated, and more can be user defined. See Appendix VIII for details. The in-built ranges are:

- ANSI – generates accented characters present in the 1252 Latin 1 codepage.
- All Latin – creates accented characters present in the Latin Unicode unicode script (0000-024F).
- Greek Extended* – generates all characters in the Greek Extended unicode script (1F00-1FFF)

To use the composite wizard:

1. Make sure you have all of the components present. Press the **details** button for details. For the Latin ranges the uppercase letters, lowercase letters and modifiers contained in the **Page Range** called **modifiers (auto)** are required.
2. Use **composite wizard** from the **glyph menu** to create the glyphs and mapping.

* The Greek Extended range is an example of user-defined generation, and can be edited (Appendix VIII), the other two ranges are hard-coded into Type 3.2.

When the wizard is complete, a report is displayed showing the number of glyphs created, the number of glyphs already present, and the number of glyphs that were not able to be created. The wizard can be run again after adding extra glyphs, if you find that some required glyphs were missing the first time.

The **composite glyph wizard** will attempt to place the diacritics in the correct position, but because of variations in font styles this operation cannot be entirely automatic. All the generated glyphs should be visually checked and manual position modifications made if required.

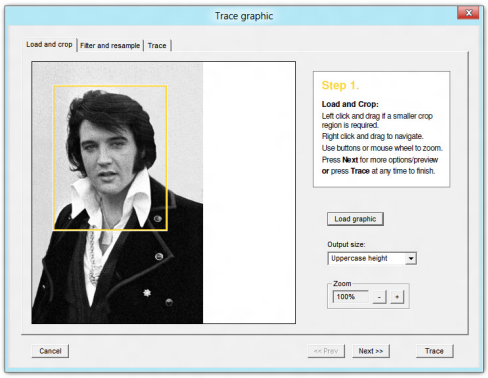
The characters that are generated by the ANSI wizard are :

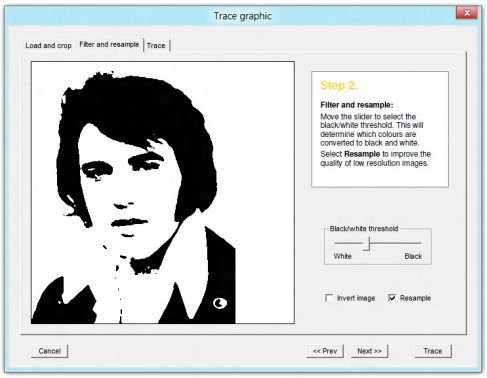
Ž Zcaron	Ë Edieresis	Ý Yacute	ï idieresis
š scaron	Ì Igrave	à agrave	ñ ntilde
ž zcaron	Í Iacute	á aacute	ò ograve
Ÿ Ydieresis	Î Icircumflex	â acircumflex	ó oacute
Š Scaron	Ï Idieresis	ã atilde	ô ocircumflex
À Agrave	Ñ Ntilde	ä adieresis	õ otilde
Á Aacute	Ò Ograve	å aring	ö odieresis
Â Acircumflex	Ó Oacute	ç ccedilla	ù ugrave
Ã Atilde	Ô Ocircumflex	è egrave	ú uacute
Ä Adieresis	Õ Otilde	é eacute	û ucircumflex
Å Aring	Ö Odieresis	ê ecircumflex	ü udieresis
Ç Ccedilla	Ù Ugrave	ë edieresis	ý yacute
È Egrave	Ú Uacute	ì igrave	ÿ ydieresis
É Eacute	Û Ucircumflex	í iacute	
Ê Ecircumflex	Ü Udieresis	î icircumflex	

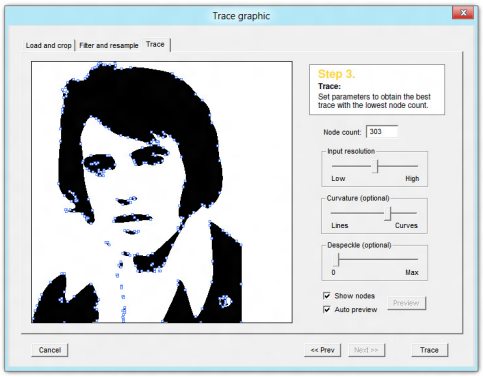
Type 3.2 allows glyphs to be created from existing graphics files. Vector graphics (.svg) can be imported directly. Raster graphics (.jpg, .gif, .png, .bmp, .tif etc) have to be converted to contours (traced), either by auto-tracing or manual tracing.

Auto Tracing (NEW to version 3.2)

Select **trace graphic**, from the **glyph menu**, or press the trace graphic button on the toolbar. The trace graphic wizard will open.







1. Press the **load graphic** button to open a graphics file. All common graphics file types are accepted (.jpg, .gif, .png, .tif, .bmp etc).

2. If you do not want to trace the entire image, then draw a crop region while pressing the left mouse button.

3. Select the desired output size for the trace from the drop down menu.

4. Press **next**. Move the slider to set the black/white threshold.

5. Press **next**. Select the tracing parameters to obtain a good trace preview. The top slider will control how many nodes are used. Try to minimize the number of nodes, while maintaining the correct shape. Use the **despeckle** slider if you need to remove small dots or 'noise' from the image.

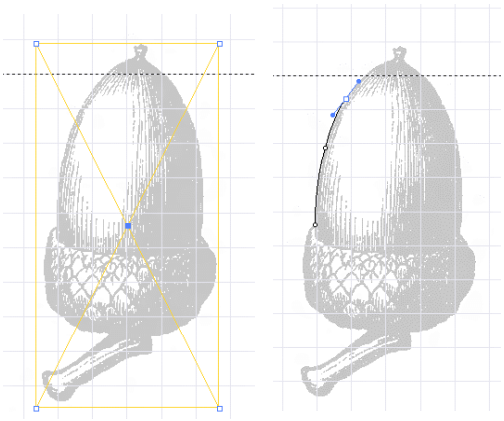
6. Press **trace** at any time to trace the image.

Manual Tracing

By inserting a graphic as a background image in the edit window, drawing tools can be used to draw outlines around the image. Select **background image** from the **view menu**, then **insert**.

To change the background image size and position double click on the image, or use the **reposition** option. Click and drag the blue corner squares to resize and the center square to move. Press **page-up** and **page-down** to alter the black/white threshold.

Double click or deselect **reposition** when the background image is in place, and the black/white threshold is correct.



Importing Vector Graphics

Vector graphics created with vector drawing applications such as Adobe Illustrator, Serif DrawPlus, CorelDRAW or Inkscape, can be imported into Type 3.2. Save the vector graphic as an **.svg** (scalable vector graphic) file. Create a blank glyph then use **import vectors**, from the **glyph menu**.

Type 3.2 will be able to import most bezier paths and basic shapes. The height or width of the graphic (which ever is largest) will be scaled to the working fonts **em unit size** (normally 1000 or 2048 units).

An Action is a set of commands that can be applied to a glyph, range of glyphs or the entire font. A simple example of an Action script is:

```
SKEWX 10
```

Which would skew a glyph 10 degrees in the x-direction. If you applied this Action to all glyphs, then you would have the beginning of an italic font.

To run an Action, press the **run action button**, or select **run action** from the **global menu**. If you want to apply an Action only to the current glyph, then you can press one of the user-assigned shortcut keys.

If the **mapping window** is in **glyph list** mode, then you can select a range of glyphs in the mapping window and they will be entered into the glyph range option when you run an Action.

Appendix VI contains a full list of commands and variables that can be used in Action scripts. Look at and try the Actions that are installed with Type 3.2 to get an idea of how they work.

For example, the **small capitals** Action script, which is designed to scale capital letters to small letter size:

`set F1 rsb`	save the current rsb in the user variable F1
`set F0 [100*xheight/caps]`	calculate the scaling as a percentage and store it in F0
`origin 0 0`	set the origin for the transformation at x=0, y=0
`stretch F0 F0`	stretch by the amount calculated in the x and y directions
`set rsb [F1*xheight/caps]`	adjust the rsb by the same proportion

Shortcut Keys

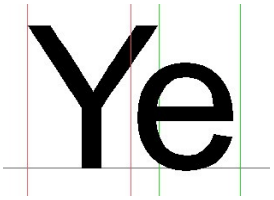
The shortcut keys 1 to 7 can be assigned to Actions. Use **assign shortcut keys** from the **global menu**. To create a new Action, select **create/edit actions** from the **global** menu, then press **new**. Enter a name for your new Action, then enter commands in the Action script window.

Composite Glyphs - Warning

Glyphs are automatically decomposed before an Action is performed but if you run an Action over a range of glyphs then the Action may end up being run on a component glyph before decomposing. This can result in unpredictable results. It is therefore is recommended that you run the **decompose** Action over the *entire font* before running Actions over large ranges of glyphs.

Certain pairs of characters look better with an altered spacing between them. Y and e, for example, look better if the e is moved closer to the Y. This is called kerning.

no kerning



with kerning

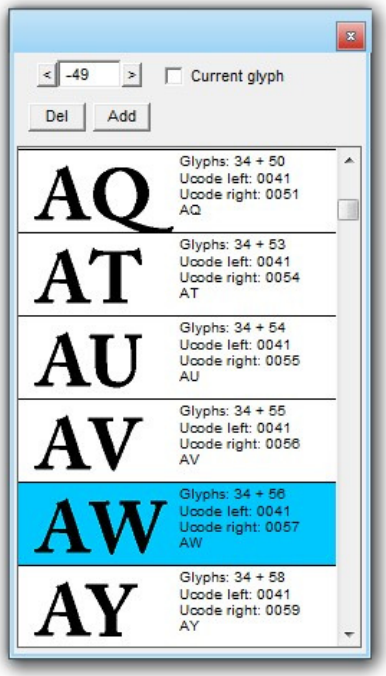
Kerning Mode

Select **kerning mode** from the **view menu** – the kerning window will appear. Uncheck **show all pairs** to show only kerning pairs that begin with the current glyph.

If you are creating a new font, you will need to add some kerning pairs. Use the **add button** to add a single pair or to add all **standard pairs** (see Appendix V). The pair(s) will be added with no initial kerning.

Click on a pair to alter the kerning - a positive value means the characters are further apart, negative is closer together. If you right click in the edit window then you can grab the right hand character to drag it into position.

Before you save your font, make sure you select **kerning** in **font options** (see Appendix I). You can also save kerning as an OpenType feature (recommended). See the section - **OpenType features**.



- A shortcut when adding a kerning pair: double click on a character in the **mapping window** (for the left-hand glyph), then single click on a character for the right-hand glyph. Now press the **add button**, and the correct unicode values will be set.
- Note that Type 3.2 will only let you kern glyphs that have been mapped to a unicode character.
- OpenType PostScript fonts (.otf) do not officially support **kern** table kerning (only OpenType feature kerning).

Not all applications will support and display kerning of fonts. To activate kerning in Microsoft Word, select your text, use **font** from Word's **format** menu, then tick **kerning for fonts** on the **character spacing** tab.

When a font is displayed at a small point size on a low resolution device (eg a screen), several distortions can occur. This happens because there are not enough pixels to accurately represent the rasterized font. Adjustments (called hints) can improve the appearance of the font at low resolutions.

Hints can either be global (apply to all glyphs within the font) or can apply to individual glyphs. The current build of Type 3.2 supports **global hinting** of **PostScript** fonts only. PostScript fonts can also be auto-hinted (automatic glyph-level hinting) with the Adobe auto-hinter (see below).

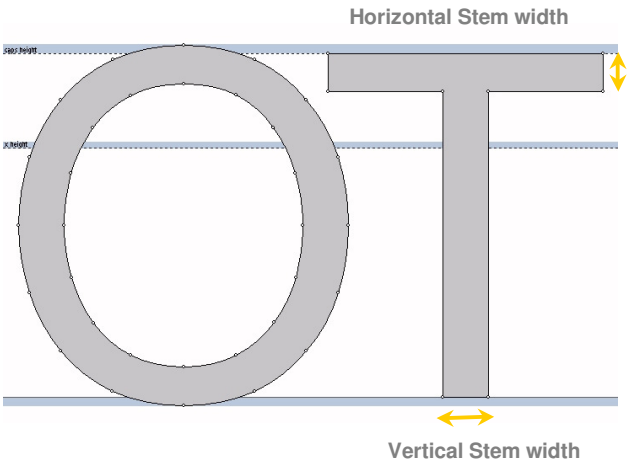
(magnified)

No hinting

Global hinting only

Global hinting and glyph-level auto-hinting

Alignment Zones



Applying global hinting to a PostScript font involves adding **alignment zones** (or **BlueValues**) and setting the values of the **stem widths**. Alignment zones are zones that you want to keep at the same height when the resolution gets very small. These will be at the baseline and typically at the top of uppercase and lowercase characters.

In the example below, curved uppercase characters (eg O) extend slightly higher (and lower) than straight uppercase characters (eg T). So zones (shown in blue) are set at the upper and lower limits.

To display alignment zones, select **alignment zones** from the **font menu**. A gray vertical bar will appear on the right side of the screen. Right click on this bar to add, edit and delete alignment zones and to set stem values.

- The first alignment zone added **must** be the **baseline zone.**
- Zones cannot overlap.
- Up to six zones above and five zones below the baseline zone are allowed.

Auto-Hinting

The **Adobe Font Development Kit for OpenType** (AFDKO) includes an auto-hinter that works with OpenType PostScript fonts. The AFDKO is a free download, available here:

www.adobe.com/devnet/opentype/afdko

It is a command line tool, which can be used in it's simplest form with the command:

```
autohint -a  myfont.otf
```

Global hinting (alignment zones and stem widths – see above) need to be added to the font before the auto-hinter can be run. Enter at least one horizontal and one vertical stem width (the standard stem widths).

ttfautohint is a project (based on the FreeType font library) for auto-hinting TrueType fonts.
A free windows GUI or command line version can be downloaded here:

www.freetype.org/ttfautohint/#download

Gasp (Grayscale) Hinting

Gasp hinting allows you to specify the rasterization technique for a TrueType font when it is rendered on grayscale-capable devices at different sizes.

A typical Gasp table could be:

Font size^	**Rasterization technique**
ppem<=8	grayscale only *
9<=ppem<=19	gridfit only **
20<=ppem	gridfit and grayscale

Select **gasp hinting** from the **font** menu. Enter 8 in the **max ppem** box, select grayscale in the **smoothing selection**, then press **add**. Enter 19 in the **max ppem** box, select gridfit in the **smoothing selection**, then press **add**. Enter 65535 in the **max ppem** box, select grayscale & gridfit in the **smoothing selection**, then press **add** (always enter 65535 as the max ppem for the last entry).

^ ppem = pixels per em. Em for a 72 point font is 1 inch (1pt = 1/72th inch).
So a 72pt font rendered on a 96 dpi (dots per inch) monitor is 96 ppem.
(So it follows that 8 ppem would be 6pt on a 96 dpi monitor).

* Note for your computer must also have font smoothing activated for grayscale rendering.
** Gridfit means use TrueType hinting instructions (they must be present in the font).

OpenType Layout Features

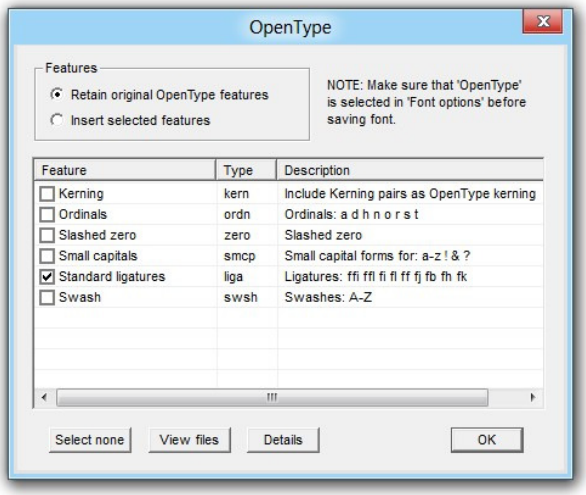
Some applications allow you to turn on an off advanced typographic features, such as ligatures, small caps and old style figures by applying formatting to the text. The features must be contained in the font as OpenType layout features. See below for a list of applications that support this type of text formatting.

OpenType feature support in Type 3.2 is not comprehensive, but supports some of the most common features types. Several OpenType feature files are installed with Type 3.2, and others can be written by the user. The syntax of Type 3.2 feature files (.feax) is similar in form Adobe OpenType feature files (.fea), but with a reduced set of commands and some other syntax differences (Appendix VII for details about how to write your own OpenType features files).

Applying OpenType features

Select **Opentype features** from the **font menu**. You will see a list of features that are currently installed.

Select a feature (small capitals for example) then press **details**. The contents of the **small capitals.feax** file will be displayed.



The line: **sub [a-z] by [A.sc-Z.sc];**

means that when **small capitals** are applied to the font, the lower case letters a-z will be replaced by glyphs named A.sc – Z.sc. For this feature to work, your font must contain the substitution glyphs named A.sc – Z.sc. (See the next section for an example of how to create these glyphs).

Assuming that the required glyphs are present (if some are missing then only the ones that are present will be functional):

1. Select the **insert selected features** button.
2. Activate the checkbox **small capitals.**
3. Select **font options** from the **font menu** and activate the retain **OpenType** checkbox.

The small capitals OpenType substitution feature will now be saved with the font.

- OpenType kerning feature requires that you generate kerning pairs (see the section **kerning**). It is recommended that you save kerning as a **kern** table and as an OpenType feature to maximize the chance that an application can support kerning with your font.

- **When using the Demo version of Type 3.2, bear in mind that only the first 50 glyphs will be saved. Any substitution glyphs after glyph 50 will not be present in the saved font.**
- Type 3.2 allows you to either add new OpenType features or to retain features already present in the font. You cannot edit the features already present in a font.

Example – Generating the 'Small Capitals' Glyphs

This example shows how you can use various features of Type 3.2 to generate the **small capitals** OpenType feature:

1. Switch the mapping window to **glyph list** view
2. Select the glyphs A-Z., right click and select **duplicate**
3. Go to the end of the glyph list and select the copies of glyphs A-Z that have been generated.
4. Right click and select **PostScript name**. Enter **A.sc**. The glyphs will now be named A.sc – Z.sc.
5. Keep the same glyphs selected, and use **run action** from the **global menu**. Select the **small capitals** Action. Select **glyph range** (should be your selected glyphs) and **run** the Action.

You now have the required glyphs for the small capitals OpenType feature - of course you can alter the substitution glyphs to suit your font – the Action is just a starting point. You might want bold them slightly for example, to match the weight of the other glyphs in the font.

Applications that Support OpenType Features

Opentype features are supported in following commercial applications:

Adobe InDesign
Adobe Photoshop
Adobe Illustrator
QuarkXPress

MS Word 2007 (only OpenType kerning is recognized)
MS Word 2010 will support some additional OpenType features

Classical Text Editor (**www.oeaw.ac.at/kvk/cte**) developed at the Austrian Academy of Sciences is a word processor that supports OpenType features. A free trial version is available.

There are some free applications that can be used to view and test your OpenType features:

TrueTypeViewer (**http://home.kabelfoon.nl/~slam/fonts/truetypeviewer.html**) is a useful tool for viewing OpenType features in action. (OpenType TrueType fonts only).

Firefox 3 supports viewing of OpenType kerning and ligatures.

OpenType Tables

Type 3.2 creates TrueType and OpenType TT fonts (.ttf) with these tables:

OS/2	**cmap**	**glyf**	**head**
hhea	**hmtx**	**loca**	**map**
name	**post**		

OpenType PS fonts (.otf) will have the above tables, but the **CFF** table replaces the **glyf** table.

When editing a font, the following tables can be included if present in the original file, or, in the case of the Gasp table, if it has been created by the user:

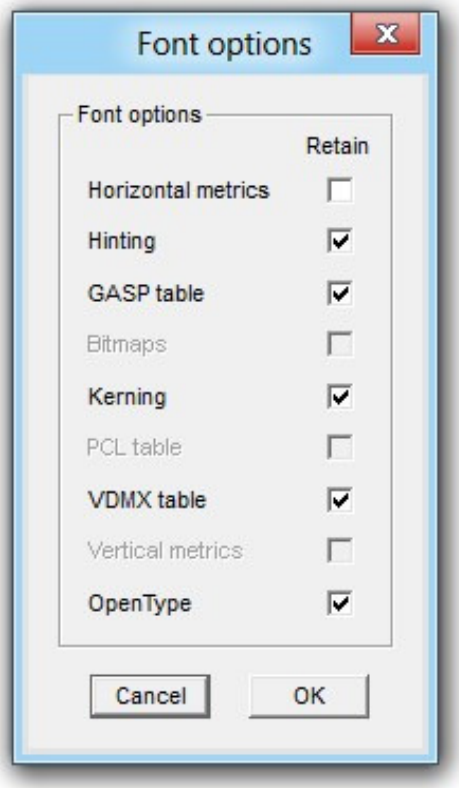
prep
cvt
LTSH
fpgm
— If 'Hinting' retained^.

EBDT
EBLC
EBSC
— If 'Bitmaps' retained.

vhea
vmtx
— If 'Vertical metrics' retained.

hdmx*
kern
gasp
PCLT**
VDMX
— Select individually.

BASE
GDEF
GPOS
GSUB
JSTF
— If 'OpenType' retained***.



^ These tables for TT fonts only. For PS fonts the hinting option will control glyph level hinting only.

* If you have edited any glyphs, then the **hdmx** (horizontal metrics) table may no longer be accurate – it is then recommended that you do not retain it. **VDMX** and **hdmx** tables can be re-created using a free tool from Microsoft called **CacheTT**. (**www.microsoft.com/typography/tools/tools.aspx)**

** If youhave changed the fonts description, then the **PCLT** table may no longer be accurate - it is then recommended that you do not retain it. The **PCLT** table is strongly discouraged for use with OpenType fonts.

*** Fonts saved as OpenType will also be given an empty **DSIG** table, This is to give the font the OpenType icon on Windows XP. The original digital signature (if any) will not be retained, as it is no longer valid for an edited font.

A Bit of History

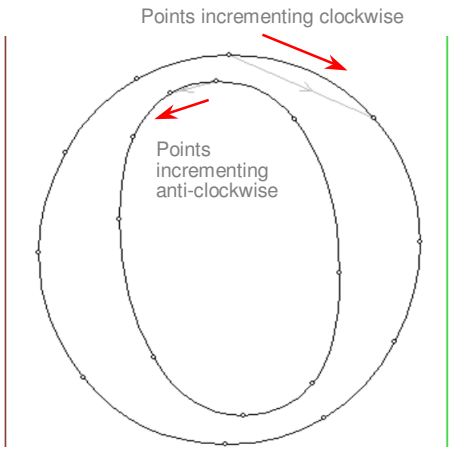
TrueType was originally developed by Apple in the late 80's, after Apple & Microsoft rejected a proposal by Adobe to use Adobe Type 1 (PostScript) fonts for their operating systems. Through a deal with Apple, TrueType was adopted by Microsoft in 1992, for their Windows 3.1 operating system.

Adobe joined forces with Microsoft in 1996 to combine their technologies and produce OpenType, which supports both TrueType and PostScript formats. Adobe finished converting its entire font library to OpenType (PostScript) fonts in 2002 with the intention that Adobe Type 1 fonts (see Appendix III) eventually be phased out.

Glyphs (TrueType)

The outlines of a TrueType glyph are defined by contours. Contours are defined by points.

Points of a contour are either on-curve (defining straight lines) or off-curve (defining a type of curve called a quadratic spline). The first point on a contour (point zero) must always be an on-curve point. See Appendix III for more details.



The filled-in area of a TrueType glyph is always on the right-hand side of the contour. So, for the letter 'o' here, the points forming the outside contour will increment clockwise, and the points forming the inner contour will increment anti-clockwise.

(Note that technically PostScript curves should be in the opposite direction - with the outside contour counterclockwise.)

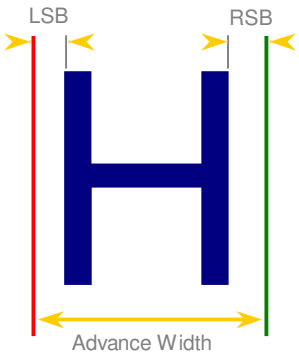
The extremes of a glyph should be defined with on-curve points.

Glyph Metrics

Certain metrics define the horizontal dimensions of a glyph. These can be set using **glyph metrics** from the **glyph menu**, or by dragging the top of the red and green vertical markers.

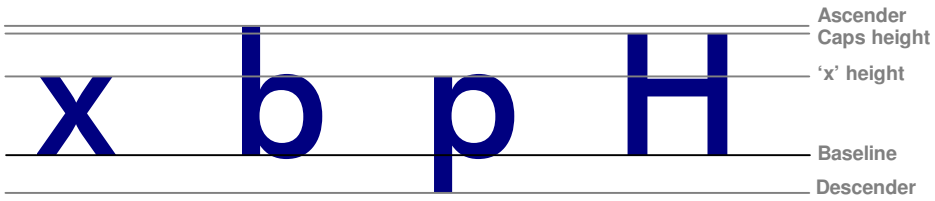
The Advance Width (AW), Left Side Bearing (LSB) and Right Side Bearing (RSB) determine the spacing between characters.

AW = LSB + character width + RSB



Font Metrics

Certain metrics define the vertical dimensions of characters in an OpenType font:



ASCENDER: Height of ascender (usually height of 'b')
DESCENDER: Height of descender (usually depth of 'p')
CAPS HEIGHT: Height of uppercase characters (defn: height of 'H')
'x' HEIGHT: Height of lowercase characters (defn: height of 'x')
BASELINE: At y=0

Do not confuse the ascender and descender with ascent and descent (called WinAscent and WinDescent in **metrics** from the **font menu**). Ascent and descent define the upper and lower limits of all glyphs – anything outside these limits will be clipped.

The **EM square** determines the size of the font when it is displayed. For example, when the font is displayed at 12 points, the EM square will be 12 points high (1 point = 1/72 inch).

The **EM square** was traditionally (from type-setting days) defined as the size of an uppercase 'M', but typically the **EM square** encompasses the ascenders and descenders with some extra (internal) leading as well.

The size of the **EM square** is usually set at 2048 units for a TrueType or OpenType TT font, and usually set at 1000 units for OpenType PS fonts.

Standard Glyphs

It is standard for a OpenType font to contain mapping to the Macintosh Roman character set, even if the font is only to be used for Microsoft Windows. Also, to meet Apple specifications, the first four glyphs of a OpenType Font should be these:

GLYPH #0 Used for **undefined** characters - normally a box shape
GLYPH #1 Special glyph with no contours, and zero width
GLYPH #2 **CR** character - no contours, but with a defined width (mapped to 0009 and 000D)
GLYPH #3 The **space** character - no contours, but with a defined width (mapped to 0020)

Visit the typography links here for more information: cr8software.net/links.html

Curves and Outlines

Outlines (the curves that form characters) are stored as a series of points, using one of two methods to mathematically describe their shape. We can say that fonts either have **PostScript outlines** or **TrueType outlines**. This appendix briefly describes the difference between the two, and the implications for font creation and editing.

Types of Outline Fonts

TrueType
extension: **.ttf**
TrueType outlines

Type 1 (not supported by Type 3.2)
extension: **.pfb** & **.pfm**
PostScript outlines.

OpenType (TT)
extension: **.ttf**
TrueType outlines. Actually identical to TrueType fonts, but may contain additional OpenType information.

OpenType (PS)
extension: **.otf**
PostScript outlines. Essentially Type 1 fonts wrapped in a TrueType file structure.

Abode's intention is that Type 1 fonts be eventually phased out and replaced by OpenType PS fonts. Adobe Type 1 fonts are not supported by Type 3.2.

TrueType and PostScript Outlines

Outlines in a font are described by a series of points. To describe a straight line, you only need to specify the co-ordinates of the two end points, but to specify a curve, you need some extra points in between. A mathematical equation called a bezier curve is used.

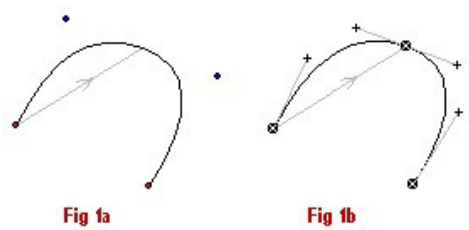


Fig. 1a shows a curve described by two end points and two off-curve points. This is a quadratic bezier, or TrueType outline.

Fig. 1b shows the same curve with three nodes and their control points. This is a cubic bezier, or PostScript outline.

In fact, quadratic beziers are a subset of cubic beziers, so any TrueType curve can be converted exactly to a PostScript one (like in Figure 1). Conversion in the other direction is not so simple, and it may require several quadratic curves to approximate a particular cubic bezier.

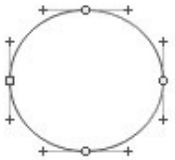


Fig 2a

Fig 2b

Fig. 2a shows a circular curve represented by a PostScript outline.

Fig. 2b, the equivalent TrueType outline, requires many more points and is only a near approximation to the original curve.

So conversion of TrueType fonts to PostScript ones is an exact science (aside from other factors such as scale and hinting), but conversion of PostScript fonts to TrueType is only a close approximation. Most commercial TrueType fonts are designed as PostScript curves, and then converted to TrueType.

Because PostScript curves are a superset of TrueType, a greater range of curves can be drawn with fewer points, making design much easier. So the cubic bezier, with its nodes and control points, is the design industry's drawing standard.

Based on an article: cr8software.net/article004.html

Appendix IV - Keyboard Shortcuts

CTRL + O	Open glyph data
CTRL + S	Save glyph data
ALT + O	Open font file
ALT + S	Save font file
ARROW KEYS	Move multiple selected points (use shift to move faster)
+ and -	Magnify and reduce glyph in edit window
CTRL + T	Toggle view mode
CTRL + Y	Toggle mapping view
CTRL + K	Toggle preview fill
CTRL + I	Toggle background image
CTRL + X	Copy whole glyph
CTRL + C	Copy selected points
CTRL + V	Paste
CTRL + B	Paste metrics only
CTRL + Z	Undo
CTRL + U	Glyph information
CTRL + J	Import vector graphic
CTRL + A	Select all
CTRL + D	Select none
ESC/Enter	Deselect point / exit reposition background image
CTRL + N	Create a new glyph
CTRL + M	Clear current glyph
CTRL + P	Point/node properties
CTRL + E	Select contour
CTRL + L	Edit actions
1-7	User-defined actions
8	Change to unlinked corner curve
9	Change to smooth curve
0	Change to symmetric curve
CTRL + R	Reverse contour
CTRL + DEL	Delete contour
DEL	Delete selected points
CTRL + F	Set size of grid
CTRL + G	Show-hide grid
F1 - F8	Select tools (1-8)
SHIFT +F1- F10	Select tools (9-18)

Appendix V - Standard Kerning Pairs

This is a list of kerning pairs* that Type 3.2 will add when using **add standard kerning pairs** from the **kerning window.**

A C	J u	T A	W ,	e w	o .	y a
A G	J ,	T O	W .	e x	p y	y e
A O	J .	T a	Y A	e y	p ,	y o
A Q	K O	T e	Y O	e ,	p .	y ,
A T	K e	T h	Y S	e .	r a	y .
A U	K o	T i	Y a	f a	r c	z e
A V	K u	T o	Y e	f e	r d	z o
A W	K y	T r	Y i	f f	r e	: (space)
A Y	L T	T u	Y o	f i	r g	, (space)
A p	L V	T w	Y u	f l	r i	, ”
A u	L W	T y	Y :	f o	r k	, ’
A v	L Y	T :	Y ;	f l	r l	. ”
A w	L y	T ;	Y -	f ”	r m	. ’
A y	L ”	T -	Y ,	f ’	r n	. (space)
A ”	L ’	T ,	Y .	f ,	r o	” A
A ’	N A	T .	a b	f .	r p	” ’
B A	N ,	U A	a g	g a	r q	’ ’
A U	N .	U ,	a p	g e	r r	’ A
B ,	O A	V A	a t	g g	r s	” (space)
B .	O T	V G	a v	g i	r t	’ ”
C A	O V	V O	a w	g o	r u	’ ’
C ,	O W	V a	a y	g r	r v	’ d
C .	O X	V e	b b	g y	r y	’ l
D A	O Y	V i	b l	g ,	r :	’ r
D V	O ,	V o	b u	g .	r ;	’ s
D W	O .	V u	b v	h y	r -	’ t
D Y	P A	V :	b y	i v	r ,	’ v
D ,	P a	V ;	b ,	k e	r .	’ (space)
D .	P e	V -	b .	k o	s w	; (space)
F A	P o	V ,	c h	k y	s ,	(space) A
F a	P ,	V .	c k	l w	s .	(space) T
F e	P .	W A	c l	l y	v a	(space) V
F i	Q U	W O	c y	m u	v e	(space) W
F o	Q ,	W a	c ,	m y	v o	(space) Y
F r	Q .	W e	c .	n u	v ,	(space) ”
F ,	R O	W h	d d	n v	v .	
F .	R T	W i	d v	n y	w a	
G ,	R U	W o	d w	o g	w e	
G .	R V	W u	d y	o v	w h	
J A	R W	W y	e b	o w	w o	
J a	R y	W :	e g	o x	w ,	
J e	S ,	W ;	e p	o y	w .	
J o	S .	W -	e v	o ,	x e	

* Based on appendix D of Adobe Technical Note # 5091.

Action scripts are text files (.atn) contained in the **My Documents/Type3files/actions** folder and consist of a series of commands entered by the user. Use **edit actions** from the **global menu** to create and edit Action files.

All commands (except for **copy**, **paste**, **selectregion** and **do**) can be applied either to a whole glyph, the selected points only (if applicable) or to the glyph contained in the clipboard, depending on the last **use** command:

use all	commands apply to the current glyph
use select	commands appy to the selected points of the glyph
use clipboard	commands appy to the clipboard glyph

Transformation and Translation Commands

move *xoffset yoffset*	move the glyph or selected points
origin *xposition yposition*	set the origin for the following transformations
rotate *angledegrees*	rotate
stretch *xpercentage ypercentage*	stretch
stroke *units*	stroke (+ or – depending on contour direction)
skewx *angledegrees*	skew in x direction (italic)
skewy *angledegrees*	skew in y direction

Variables

set variable *value*	set the stated variable to a certain value

Parameters in italics above are values. Values can be:

Variables: **rotate F0**
Numbers: **rotate 90**
Expressions: **rotate [F1*F0/4]** (*)

Expressions must be enclosed by [ ] square brackets. Parentheses ( ) are cannot be used to group calculations in expressions. So to get:

```
F1 = (xmax-minx)/2
```

you need to use an intermediate step:

```
set F0 [xmax-xmin]
Set F1 [F0/2]
```

Variable List

The following are a list of valid variables. Note that those that are font parameters, rather than glyph-specific parameters, are read only (eg **x-height**). Glyph specific variables (eg **xmin**) will relate to the current glyph or the clipboard glyph, depending on the last **use** command.

rsb	right side bearing
lsb	left side bearing
p1	red marker
p2	green marker
aw	advance width*
xmin	x value of left-most point**
xmax	x value of right-most point**
ymin	y value of lowest point**
ymax	y value of highest point**
F0 – F9	user defined variables (can only contain integer values)
gap	used for composite wizard
xheight	x-height (read only)
caps	caps height (read only)
em	em unit size (read only)
rand*value*	random number between 0 and value eg **rand100** (read only)

* The advance width cannot be changed using the **set** command if the font is monospaced.
** Altering these will move the whole glyph so that the condition is satisfied. Eg **set xmin 0** will move the whole glyph so that the leftmost point lies on x=0.

Other Commands

do "action"	run another action
load "filename.gfs" *glyphindex*	load a glyph from a gfs file.. Can be path and filename or just filename if the .gfs file is located in the actions folder.
selectregion *x1 y1 x2 y2*	select all points contained in a rectangular region
clear	clear the current glyph, clipboard or selected points
copy	copy the current glyph to the clipboard
copyselect	copy selected points to the clipboard
paste	paste the clipboard to the current glyph (merge)
smooth	smooth whole glyph or selected points
straighten	straighten whole glyph or selected points
reverse	reverse direction of all or selected contours
truetype	convert glyph to TrueType curves after running action*
decompose	decompose glyph**

* Default output of an Action is a PostScript glyph. Use **truetype** command to convert to TrueType.
** The **decompose** command is redundant because glyphs are decomposed before running any action.

Type 3.2 OpenType feature files (.feax) are located in the **My Documents/Type3files/opentype** folder. Select **OpenType features** from the **font menu**, then press **view files** to open this folder. Files can be edited using Windows notepad (double clicking on the file will open it in notepad).

A simple example is (**slashed zero.feax**):

```
languagesystem DFLT dflt;
languagesystem latn dflt;

feature zero {
    sub zero by zero.slashed;
} zero;
```

In the above example, when the feature tag called **zero** is activated, the zero character will be replaced by the glyph named **zero.slashed** (zero with a slash through it). This feature applies for the specified script (latn - latin) and language (dflt – default). Using **`languagesystem DFLT dflt`** as the first entry (recommended) also means that the feature will apply if no language/script match is found.

Each **.feax** file can contain more than one feature block (**`feature <tag>{...}`**) - see **mixing lookup types** below - but each block must be for the same feature tag *(this differs from Adobe feature files).*

Lookup Types

Type 3.2 supports the following lookup types:

Single substitution	**`sub A by B`**	replace A with B
Multiple substitution	**`sub A by B C D`**	replace A with the sequence BCD
Alternate substitution	**`sub A by A1 A2 A3`**	replace A with one of A1 A2 or A3
Ligature substitution	**`sub A B by C`**	replace the sequence AB with C
Single adjustment positioning	**`pos A <value>`**	adjust the position of A by <value>
Pair adjustment positioning	**`pos A B <value>`**	for the pair AB adjust the position of A by <value>

*The words **`substitution`** and **`position`** can be used instead of **`sub`** and **`pos`**.

Position adjustment lookups use a position adjustment value of the form:

`< dx dy dAWx dAWy >` or just **`dAWx`**

where **`dx`** is the x position adjustment, **`dy`** is the y position adjustment, **`dAWx`** is the horizontal Advance Width adjustment and **`dAWy`** is the vertical Advance Width adjustment.

Adjustment value examples:

`pos A Y <0 0 –10 0>` and **`pos A Y –10`**

are both equivalent: adjust the advance width of A by –10 when it is before a Y (a kerning pair).

`pos A <0 10 0 0>`

will move the glyph A up by 10 units. (y=+10)

Note: Classes are not supported by Type 3.2 features.

Mixing Lookup Types

A feature can make use of more that one lookup type. **Each lookup type must have a separate feature block.** *(Type 3.2 feature files differ from Adobe in this respect).*

You could use the following, for example, to implement old style figures:

```
languagesystem DFLT dflt;
languagesystem latn dflt;

feature onum {
      sub zero by zero.os;
      sub one by one.os;
      sub two by two.os;
} onum;

feature onum {
      pos three <0 –200 0 0>
      pos four <0 –200 0 0>
      pos five <0 –200 0 0>
      pos seven <0 –200 0 0>
      pos nine <0 –200 0 0>
} onum;
```

Substitution glyphs are used to replace 1 2 and 3 with characters that are lowercase height, but position adjustments are used to lower the 3 4 5 7 and 9 characters to the x-height. The actual value of y adjustment used will depend on the font's metrics and design – in this case a y adjustment value of –200 is used. This is for example - in practice you could use substitution glyphs for all characters if you also wanted different shapes for the dropped characters.

Xx 0123456789 (oldstyle figures – X's for size comparison)

GDEF (Glyph Definition) Table

You can classify certain glyphs so that certain features will not be applied to them. This glyph classification is stored in a GDEF table. Allowed classifications are: **baseglyphs**, **ligatures**, **marks** and **components**.

For example in the **ligature.fea** *(this differs from Adobe syntax):*

```
gdef {
baseglyphs default
ligatures f_f_i f_f_l f_i f_l;
ligatures f_f f_j f_b f_h f_k;
} gdef;
```

classifies the listed glyphs as **ligatures** and all others as **baseglyphs**.

You may have a feature, eg small capitals that you don't want applied to ligatures.
Use the command **`ignore ligatures`** as the final entry in the feature block:

```
feature smcp {
      sub [a-z] by [A.sc-Z.sc];
      ignore ligatures;
} smcp;
```

For this to work, the ligatures must be applied to the text *before* the small capitals. OpenType features are applied in the order that they are listed in **OpenType features** from the **font menu**. To make sure that the **liga** feature is applied first, do this:

1. Select **OpenType features** from the **font menu**, select *only* **ligatures** then press **OK.**
2. Select **OpenType features** again (**liga** will now be at the top), select **small capitals.** Press **OK.**

Adobe AFDKO

Adobe provides a command line compiler for OpenType feature files (.fea) as part of their AFDKO. It is a more comprehensive (and more complex) than Type 3.2:

www.adobe.com/devnet/opentype/afdko/
www.adobe.com/devnet/opentype/afdko/topic_feature_file_syntax.html (documentation)

Composite Wizard files (.cwz) are located in the **My Documents/Type3files/composites** folder. They are plain text files that can be edited or added using Windows Notepad. A new install of Type 3.2 contains only **Greek Extended.cwz**, and this will be used as an example to explain the syntax and operation of .cwz files.

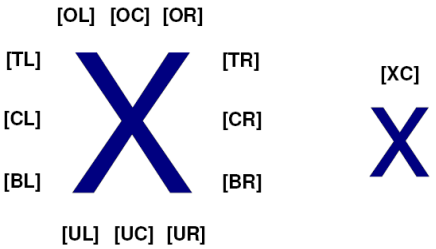
To create a composite: a **base glyph**, **modifier glyph**, **resulting glyph** and a **positioning flag** must be specified in the form:

```
1F08 = 0391 1FBF [TL]
```

The unicode character 1F08 will be generated from the base glyph 0391 and modifier glyph 1FBF. The modifier will be placed at the **T**op **L**eft of the base glyph.

Allowed positioning flags are shown here. If no flag is specified, then the default position is `[OC]` – **O**ver **C**entre.

Also allowed are `[ZL]` and `[ZR]` which simply places the modifier glyph to the left or right of the base glyph without adjusting its vertical position.



The distance between the modifier and the extremes of the base glyph is set by a variable called `GAP`. (Except for `[XC]` where it is the distance between the modifier and the x-height). The value of `GAP` can be set and changed at any time using the `SET` command (see Appendix VI). `SET` is the only Action command that can be used in a .cwz file.

In the Greek Extended example, the first command is: `SET F0 [CAPS/10]`

This sets the user variable `F0` to one-tenth the Caps Height. Later in the script there are commands that change the value of the `GAP` variable before generating other composites:

```
SET GAP F0
SET GAP [-2*F0]
SET GAP 0
```

In some cases an existing glyph may need to be mapped to another unicode character:

```
1FEF = 0300
```

The glyph mapped to unicode 0300 will now also be mapped to unicode 1FEF.

If you want gylphs to be decomposed as they are created, the commands `DECOMPOSE ON` and `DECOMPOSE OFF` can be used. In the Greek Extended example, modifiers are decomposed.

Lines of text that start with the **#** character are comments, and will be displayed when the user presses the **details** button. This function is used to list the required glyphs.

LICENSE AGREEMENT

Type 3.2 font editor is Copyright ©2006-2012 CR8 Software Solutions ("Software Publisher").
All rights reserved.

This End User License Agreement accompanies the **Type 3.2 font editor** ("Software") and applies to all associated files (except third party fonts and FreeImage library), any upgrades, documentation, modified versions or updates of the Software.

1. USE OF THE SOFTWARE.

You may use a single installation of the Software product for the production of fonts for personal or commercial use. Installation of the Software on multiple machines requires a seperate license for each installation.

2. COPYRIGHT.

The Software is copyright of the Software Publisher. You may not copy, sell or redistribute the Software. This Agreement does not grant you any intellectual property rights in The Software.

3. RESTRICTIONS.

You agree not to modify, adapt, translate, reverse engineer, decompile, disassemble or otherwise attempt to discover the source code of the Software. You may not alter or modify the installer program or create a new installer for the Software.

4. LIMITED WARRANTY

In no event will the Software Publisher be liable for indirect, special, incidental, tort, economic, cover or consequential damages arising out of the use of or inability to use the Software, including, without limitation, damages or costs relating to the loss of profits, business, goodwill, data or computer programs, even if advised of the possibility of such damages. In no case shall the Software Publisher be liable for money damages exceed the amount paid by you for the Software out of which such claim arose. The Software Publisher limits liability, according to the terms of this Agreement, to the extent permissible at law.

5. THIRD PARTY COMPONENTS

This End User License Agreement does not apply to certain third party fonts bundled with this package. These fonts, and their respective licenses, are contained in the /enc subdirectory of the install directory.

The FeeImage graphics library (FreeImage.dll) is used and distributed under the FreeImage Public License 1.0. See freeimage-license.txt in the install directory.

Potrace Professional (TM) 1.0r is Copyright (C) 2001-2012 Icosasoft Software Inc. All rights reserved. Potrace Professional is used under license from Icosasoft Software Inc. typetracer.dll may only be used with **Type 3.2 font editor**, and may only be distributed within the **Type 3 font editor** install program.

YOUR ACCEPTANCE OF THE FOREGOING AGREEMENT WAS INDICATED DURING INSTALLATION.

© 2006 - 2012 Allan Murray
CR8 Software Solutions
cr8software.net

260512033